

CCS64 V4.1

(C) 1995 - 2026 PER HÅKAN SUNDELL.

(C) 1995 - 2026 COMPUTERBRAINS.

All rights reserved.

27 July 2026

(C) 1995 - 2026 PER HÅKAN SUNDELL.
(C) 1995 - 2026 COMPUTERBRAINS.
All rights reserved.

Contents

1	Introduction	5
2	Quick Start	7
2.1	Running your first program	7
3	Installation	9
3.1	Windows requirements	9
3.1.1	Supported operating systems:	9
3.1.2	Hardware requirements:	9
3.1.3	Additional support:	9
3.2	Portable installation	9
3.3	Configuration files	10
3.4	ROM files	10
3.5	32-bit vs 64-bit differences	10
4	Basic Usage	11
4.1	Menu navigation	11
4.2	Keyboard mapping	12
4.2.1	Insert C64 Key Stroke	12
4.3	Loading files	12
4.3.1	Programs	13
4.3.2	Tape	13
4.3.3	Disk	15
4.3.4	Cartridge	16
4.4	Joystick and Input setup	17
4.4.1	Automatic Joystick Detection	18
4.5	Display and Audio setup	19
4.6	Speed Control and Pause mode	19
4.6.1	Speed-o-meter	20
4.7	Saving states	20
4.8	Screenshots and Audio/Video Recording	20
5	Supported Hardware and Emulation	23
5.1	CPU	23
5.2	VIC-II	24

5.3	SID	25
5.4	CIA	25
5.5	1541 Disk Drive	26
5.6	Tape	27
5.7	Cartridges	27
5.8	REU	28
5.9	Printers	29
5.10	Input devices	29
5.11	Displays	30
6	Advanced Features	33
6.1	Rewind/Replay	33
6.2	Reverse Execution	34
6.3	Kaillera Multi-play	34
7	Machine-Code Monitor	37
7.1	Starting	37
7.2	Command Overview	38
7.3	Guides	40
	7.3.1 Coding	40
	7.3.2 Memory	40
	7.3.3 Debugging	41
8	Command-Line Reference	43
9	Configuration Reference	45
10	Troubleshooting	47
11	Technical Notes	49
12	Contact / Project Links	51

Chapter 1

Introduction



Figure 1.1: A typical setup of the Commodore 64 home computer system.

CCS64 is a high-accuracy emulator of the Commodore 64 home computer system and its associated hardware, including the 1541 disk drive, cartridge hardware, cassette devices, and peripheral equipment, see Figure 1.1. The emulator is designed to reproduce the original behavior of the Commodore 64 at the hardware level, allowing commercial games, demos, utilities, and timing-sensitive software to operate as they did on the original machines. Beyond basic compatibility, CCS64 focuses on accurate emulation of internal hardware timing, device interaction, graphics generation, audio synthesis, and low-level system behavior.

Development of CCS64 began in 1995 during the early years of Commodore emulation on personal computers. From the beginning, the project pursued a different direction than many contemporary emulators by emphasizing hardware research, cycle-exact execution, and faithful reproduction of undocumented behavior. Over the years, CCS64 evolved from a DOS-based emulator into a modern Windows application supporting advanced graphics rendering, audio emulation, replay functionality, debugging tools, multiplayer support, and extensive cartridge and disk-drive emulation. The project has continuously incorporated knowledge gathered from hardware testing, reverse engineering, and the Commodore demo and preservation communities.

The philosophy behind CCS64 is based on preservation, compatibility, and hardware accuracy. Rather than approximating the behavior of the original machine, CCS64 attempts to emulate the internal operation of the Commodore 64 and its peripherals as closely as possible, including many hardware side-effects and timing details relied upon by advanced software productions. This approach allows the emulator to support a wide range of software that depends upon exact CPU timing, VIC-II video behavior, disk-drive synchronization, or SID sound characteristics. CCS64 is therefore intended not only as a platform for playing classic software, but also as a tool for digital preservation, technical study, software development, and historical research into Commodore 64 hardware and software systems.

Chapter 2

Quick Start

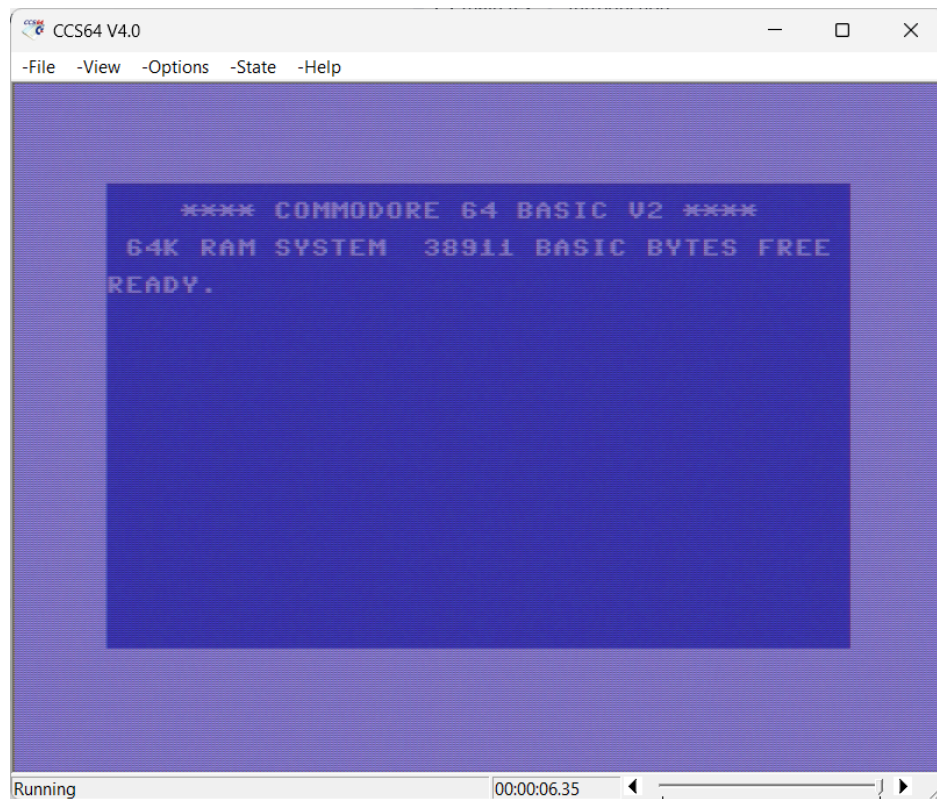


Figure 2.1: CCS64 running in windowed mode.

2.1 Running your first program

1. Start CCS64. After starting the program executable, the emulator typically starts up in windowed mode and starts emulating the Commodore 64 including its builtin operating system (KERNAL) and the associated BASIC interpreter, see Figure 2.1.
2. Select "Load and Run..." in the "File" Windows menu. This will present you with a

file dialog where you can select your previously downloaded C64 files (e.g. having a filename ending with .T64 or .D64). After selecting the C64 file, the emulator will restart and reset the Commodore 64 system. It will thereafter automatically load the selected C64 file and start it (e.g. by automatically typing "RUN").

3. The emulation has now started. Follow the action on the screen and eventually your program has started. Interact with the emulated program using the keyboard. If the programs expects input from a joystick, you can use the pre-mapped keys on the numeric keyboard. Press Alt+F10 to quickly swap joystick ports if the program requires the other port.

Chapter 3

Installation

CCS64 is currently distributed as a 32-bit or 64-bit version for the Microsoft Windows operating system.

3.1 Windows requirements

3.1.1 Supported operating systems:

- Windows 7/8/9/10/11 with DirectX v9.0 or later.

3.1.2 Hardware requirements:

- Pentium Dual-Core processor or better.
- 256 MB of internal memory or more.
- SXGA graphics or better, supporting DirectX version 9 or later.

3.1.3 Additional support:

- Windows compatible sound card.
- Windows compatible joystick.
- Windows compatible mouse.
- HardSID sound add-on card. (Note: only available with the 32-bit version)

3.2 Portable installation

CCS64 can be either installed on the Windows system by using the supplied installer package or be run directly without installing by executing the main executable file.

3.3 Configuration files

If you have any problems with the settings and want the factory-default settings back, just delete the 'C64.CFG' configuration file in your CCS64 installation folder, and it will be re-stored back to default values, next time you run CCS64.

3.4 ROM files

The emulator supports configuration to use different ROM sets than the original.

The C64 and 1541 ROMs are (C) Commodore Business Machines. If you don't want these original ROMs, you should replace the files KERNAL.ROM, BASIC.ROM, C1541.ROM, and CHAR.ROM with alternative ROM images.

3.5 32-bit vs 64-bit differences

HardSID support is only available with the 32-bit version.

Supraclient is used as a replacement for the Kaillera client with the 64-bit version.

Chapter 4

Basic Usage

The emulator provides two user interfaces:

- Text-based User Interface (TUI). This is designed in a C64 style with the aim to look as it is running on a C64.
- Graphical User Interface (GUI). This is the standard Windows interface with buttons, sliders and pop-up menu systems.

The TUI is the main user interface and is always available. The GUI is only available in windowed mode; full-screen mode uses the TUI exclusively. The menu system within the TUI is the main menu system and has all options within it, and is here-after referred to as "the menu".

4.1 Menu navigation

To use the menu, press the [F9] key, and then use the cursor keys to select different options. Use the [Cursor Right] or [Enter] keys to enter a choice. Use the [Escape] key to go back to the previous menu from within a sub-menu. Use the [Cursor Up/Down] keys to change the selection. Use the [Cursor Left/Right] keys to change a value. As specified at the bottom of the menu, you could also use the corresponding function keys. When a selection can be applied, the [Enter] key can also be used.

In a directory listing, you could also use the [Page Up/Down] keys to scroll faster through the listing. To get directly to the first file-name, which starts with a specific letter, you can press [Ctrl] + [Letter].

In some (sub-)menus, you could also select an option by pressing the first letter in the option name, like [S] for the option "Select Disk".

The user interface uses the following keys during the emulation session:

F9	The menu. Always continues from the sub-menu where you left.
F10	The menu.
F11	Restores a frozen emulator state with the filename "freeze.c64".
F12	Freezes (saves) the emulator state with the filename "freeze.c64".

The user interface also uses a set of fast-keys that are active during the emulation session and within the menu system, see Table 4.1.

4.2 Keyboard mapping

The keyboard is emulated almost like a real C64 keyboard and follows the C64 keyboard layout, except for these keys:

C64 key	Emulation key
 Commodore	[Left CTRL]
 RUN/STOP	[Escape]
 RESTORE	[Page Up]. Observe that you must HIT this key like on a real C64!
 CLR/HOME	[Home]
 Arrow Up	[Delete]
 Pound	[Insert]
 Control	[Tab]

4.2.1 Insert C64 Key Stroke

This useful feature allows you to easily type anything that you could have typed on a real C64 keyboard. Many users find it hard sometimes to find a particular C64 character or C64 character combination, because these may reside on a completely different place on the PC keyboard. By pressing [Alt] + [K], then using the cursors keys, you can choose between every possible C64 key-stroke, press the [Enter] or [Space] keys, and then the key-stroke will be automatically typed. This saves you pressing long-winded combinations of keys on the PC keyboard, in order to emulate the pressing of a special key on the equivalent C64 keyboard.

4.3 Loading files

The Commodore 64 ecosystem originally relied upon a variety of external hardware devices and storage media for loading software and data. Programs were commonly distributed on cassette tapes using the Commodore C2N tape system, on floppy disks using devices such as the Commodore 1541 disk drive, or through ROM cartridges connected directly to the Commodore 64 expansion port. Each of these storage systems possessed distinct electrical, mechanical, and timing-related characteristics that formed an integral part of how Commodore 64 software operated.

Today, the original hardware devices and storage media have become increasingly difficult to use with modern computer systems. Cassette recorders, floppy drives, and cartridge

hardware typically require specialized adapters or custom-built interfaces in order to communicate with contemporary PCs. In addition, the original storage media themselves are often several decades old and may suffer from physical degradation, magnetic signal loss, mechanical wear, or hardware failure. As a result, preserving both the software contents and the behavior of the original hardware systems has become an important aspect of Commodore 64 emulation and digital preservation.

CCS64 therefore emulates not only the Commodore 64 itself, but also the associated storage devices and media in the form of file images. These file images represent either the logical contents or the low-level physical structure of the original media and can be loaded into the emulator as virtual tapes, floppy disks, or cartridges. Depending on the file format used, the image may preserve anything from ordinary program data to detailed timing information, disk encoding patterns, tape pulse sequences, or cartridge banking behavior. Through this approach, CCS64 can reproduce much of the operational behavior of original Commodore 64 hardware systems while simultaneously contributing to long-term software preservation and accessibility on modern computer platforms.

CCS64 currently supports the following file image formats: .PRG, .P00, .T64, .TAP, .D64, .G41, .G64 and .CRT. When running in windowed mode, CCS64 supports drag-and-drop, which automatically performs a default load/run of the image file dropped. For any valid Commodore 64 file format that is associated with CCS64 in the Windows Registry, double-clicking on that Commodore 64 format file will automatically start it within CCS64. When loading image files using the menu system, you typically first have to select the device (1541 disk drive, tape, or cartridge) through which the loading will be done and then select the corresponding file. See Chapter 5 for a detailed description of the hardware emulated by each storage device.

4.3.1 Programs

Loading individual programs and data directly into the C64's memory is supported by .PRG files. See Figure 4.1 for a graphical description of the .PRG file format. The emulator also supports loading from the similar .P00 file format.

The default device through which the .PRG and .P00 files are loaded is the first 1541 disk drive. However, the actual loading is typically handled as a "fast-load", i.e., the kernal call is intercepted and the transfer to memory is handled by the emulator natively instead of emulating the device.

4.3.2 Tape

Loading programs and data from cassette tapes are supported by either .T64 archive files or by .TAP raw-tape files. See Figure 4.2 and Figure 4.3 for a graphical description of the .T64 and .TAP file formats respectively.

When loading .T64 and .TAP files using the menu system, you typically first have to select "Tape Device 1..." and then select the corresponding file. However, the actual loading of .T64 is typically handled as a "fast-load", i.e., the kernal call is intercepted and the transfer to memory is handled by the emulator natively instead of emulating the device. There is an option for loading the file as "Turbo Tape" that will cause the device to be emulated.

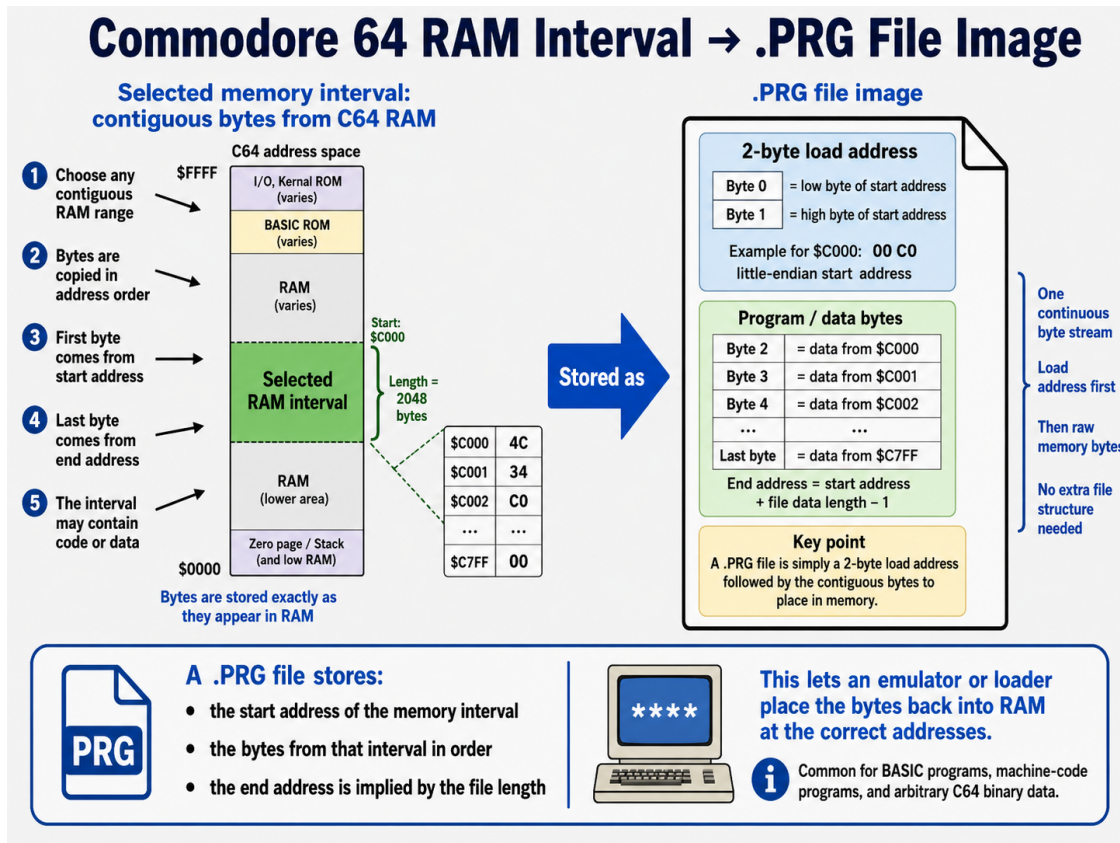


Figure 4.1: Illustrating the relation between C64 memory contents and the .PRG file image.

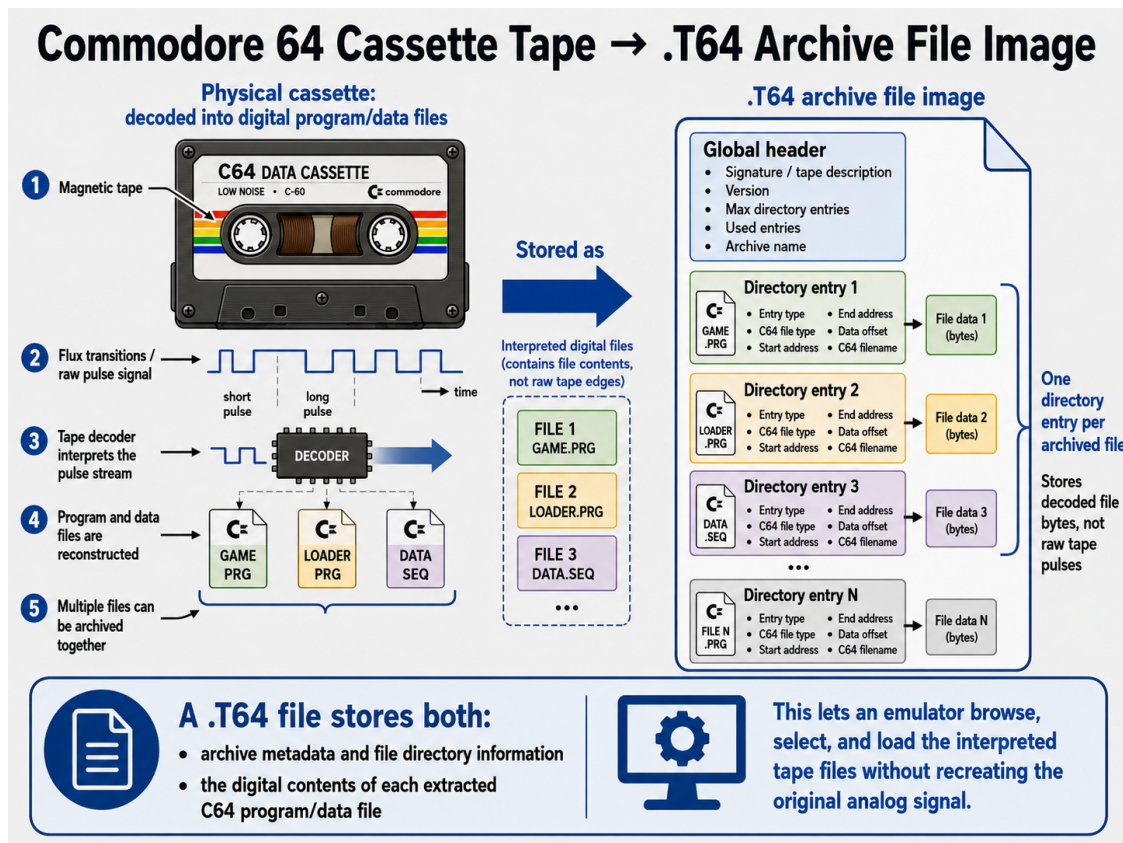
C2N Status

To enable an overlay display, continuously showing the current tape position (as C64 cycles counted from the start), select the corresponding option in the "Special..." sub-menu.

Tape Drive Noise

You can listen to the tape drive noise that would be generated whilst loading raw-tape (.TAP) files. Simply perform the following steps:

1. In the "Special..." sub-menu, ensure that the "C2N Noise" option is set to On.
2. Type POKE 54296,15 and press [Enter] from BASIC, in order to set the SID volume to the maximum.
3. Type LOAD and press [Enter] from BASIC, so that the loading from device 1 is initiated from the BASIC interpreter.
4. Select the desired raw-tape (.TAP) file, by using the option "Select Tape" in the "Tape Device 1..." sub-menu.



A .T64 file stores both:

- archive metadata and file directory information
- the digital contents of each extracted C64 program/data file

This lets an emulator browse, select, and load the interpreted tape files without recreating the original analog signal.

Figure 4.2: Illustrating the relation between C64 Cassette Tapes and the .T64 archive file image.

4.3.3 Disk

Loading programs and data from floppy disks is supported by either .D64 files or by .G64 (also sometimes named .G41) files. See Figure 4.4 and Figure 4.5 for a graphical description of the .D64 and .G64 file formats respectively.

When loading .D64 and .G64 files using the menu system, you typically first have to select "1541 Device 8..." and then select the corresponding file. There are options to choose whether the file is handled as "fast-load" (i.e., the kernal call is intercepted and the transfer to memory is handled by the emulator natively instead of emulating the device) or "normal" (the device is fully emulated). Note that some program might not work correctly if the loading is handled as "fast-load", and thus requires "normal" handling using full emulation of the device. See Section 5.5 for details of the 1541 drive emulation.

Maximum 1541 Speed

However, when fully emulating the device, loading takes the same amount of time as with a real disk drive, which can mean several minutes. Therefore, the option "Maximum 1541 Speed" can be enabled on the "Special..." sub-menu. This option will speed up emulation to a maximum while the loading takes place and thus can reduce the loading time to a few

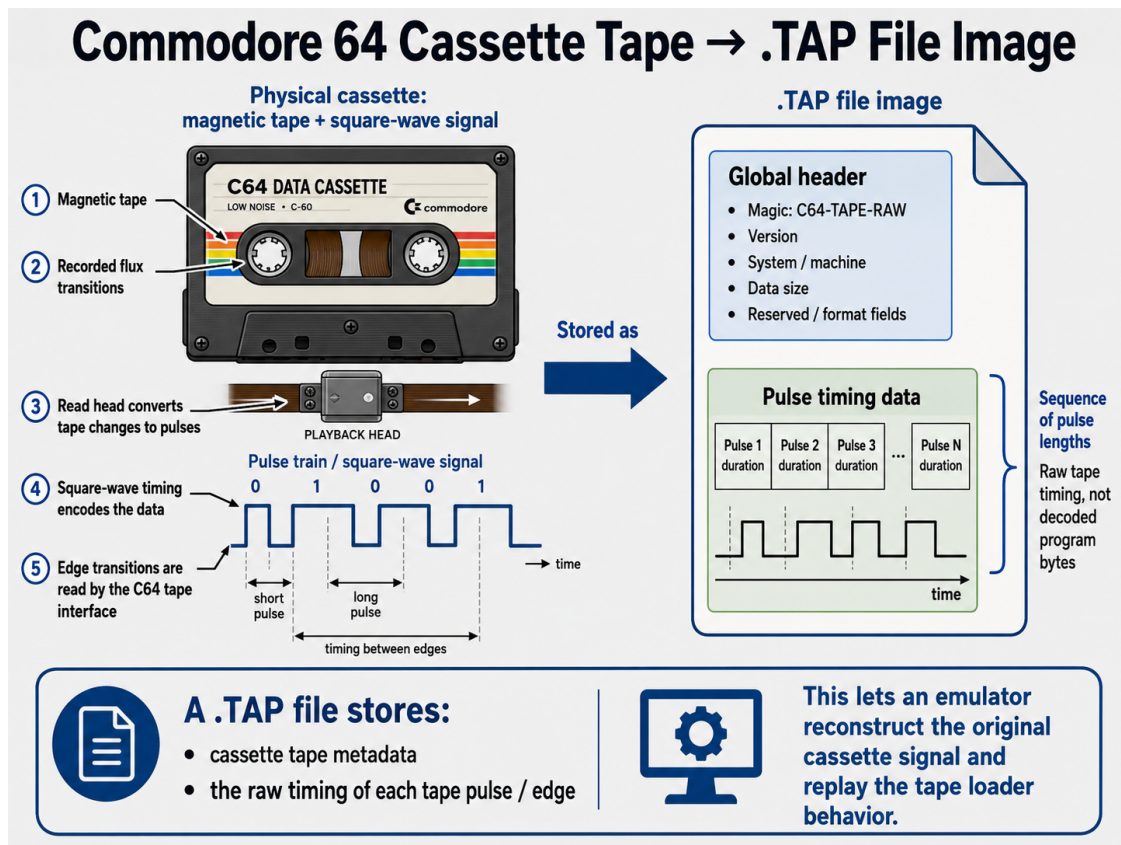


Figure 4.3: Illustrating the relation between the C64 Cassette Tapes and the .TAP raw-tape file image.

seconds.

C1541 Status

To enable an overlay display, continuously showing the current head position (as track number) and r/w mode of up to four active 1541 disk drives, select the corresponding option in the "Special..." sub-menu.

4.3.4 Cartridge

Cartridges can also hold programs and data contents as well as additional hardware related functionality. The emulator supports the insertion of external cartridges by the means of .CRT files. See Figure 4.6 for a graphical description of the .CRT file format,

When loading (i.e., inserting a cartridge into the expansion port of the Commodore 64) .CRT files using the menu system, you typically first have to open the "Cartridge..." sub-menu and select "Insert Cartridge...", and then finally select the corresponding file. To remove the cartridge, use the same sub-menu and select "Eject Cartridge".

Some cartridges have a builtin hardware button for "freezing". This button is also emulated and can be triggered during emulation by pressing [Alt]+[F] or selecting the corre-

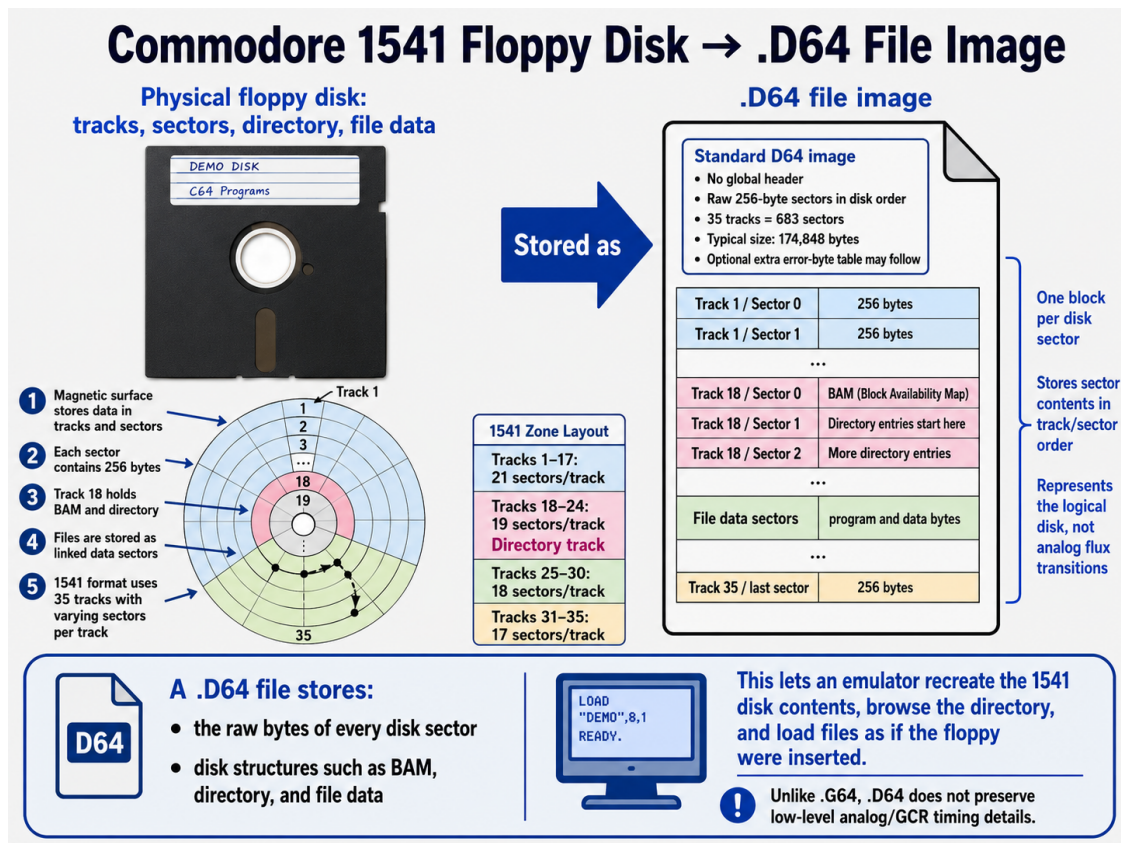


Figure 4.4: Illustrating the relation between C1541 Floppy Disks and the .D64 archive file image.

sponding menu choice. See Section 5.7 for cartridge architecture and memory mapping.

4.4 Joystick and Input setup

CCS64 can emulate the original Commodore joysticks through either:

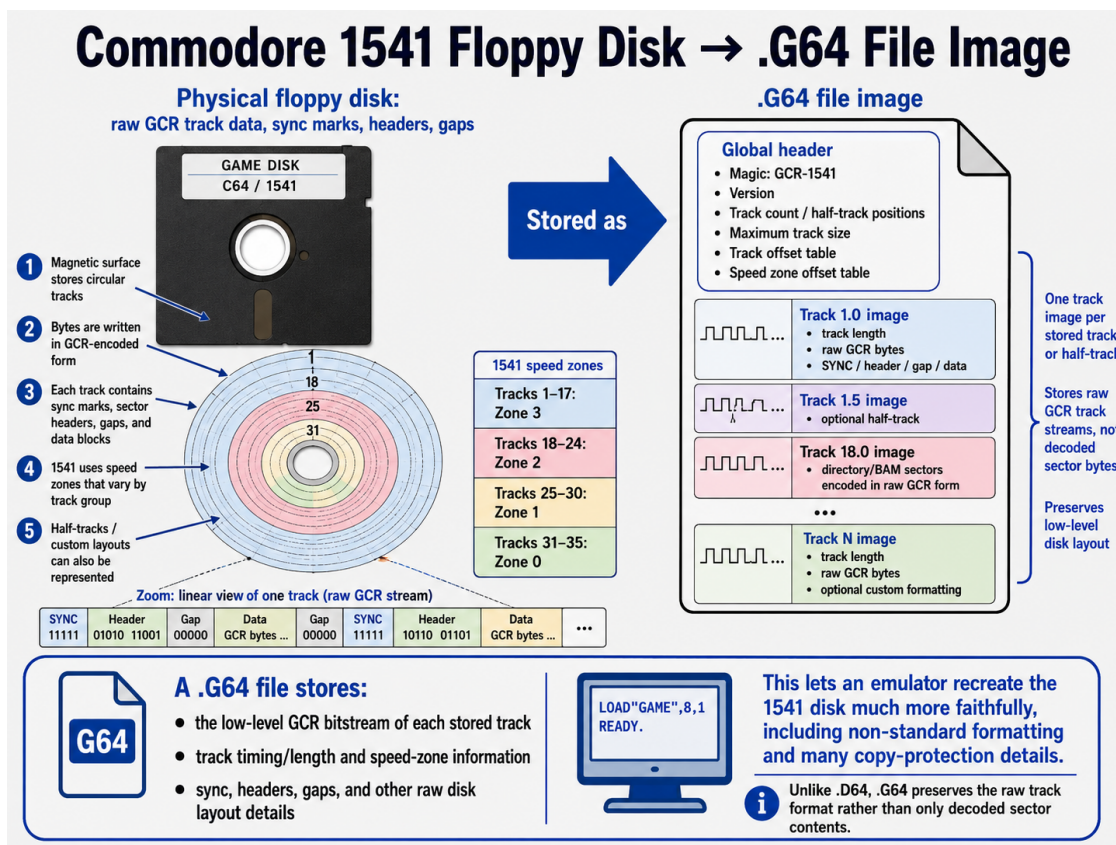
- modern video game controllers (e.g. gamepads or joysticks) connected to the PC.
- mapping of certain keys on the PC keyboard to the corresponding joystick inputs.

By opening the "Input..." sub-menu, you can select the appropriate type of emulation for the corresponding joystick port. Note that for the best experience with using PC joysticks, they should be calibrated within the emulator using the "Calibrate Joysticks..." option.

The default key-sets for joystick emulation with keys is defined as follows:

C64 Joystick Port 1

Up	[Keypad %]
Down	[Keypad 5]
Left	[Keypad 7]
Right	[Keypad 9]
Fire button 1	[Right Alt] ([Alt Gr])



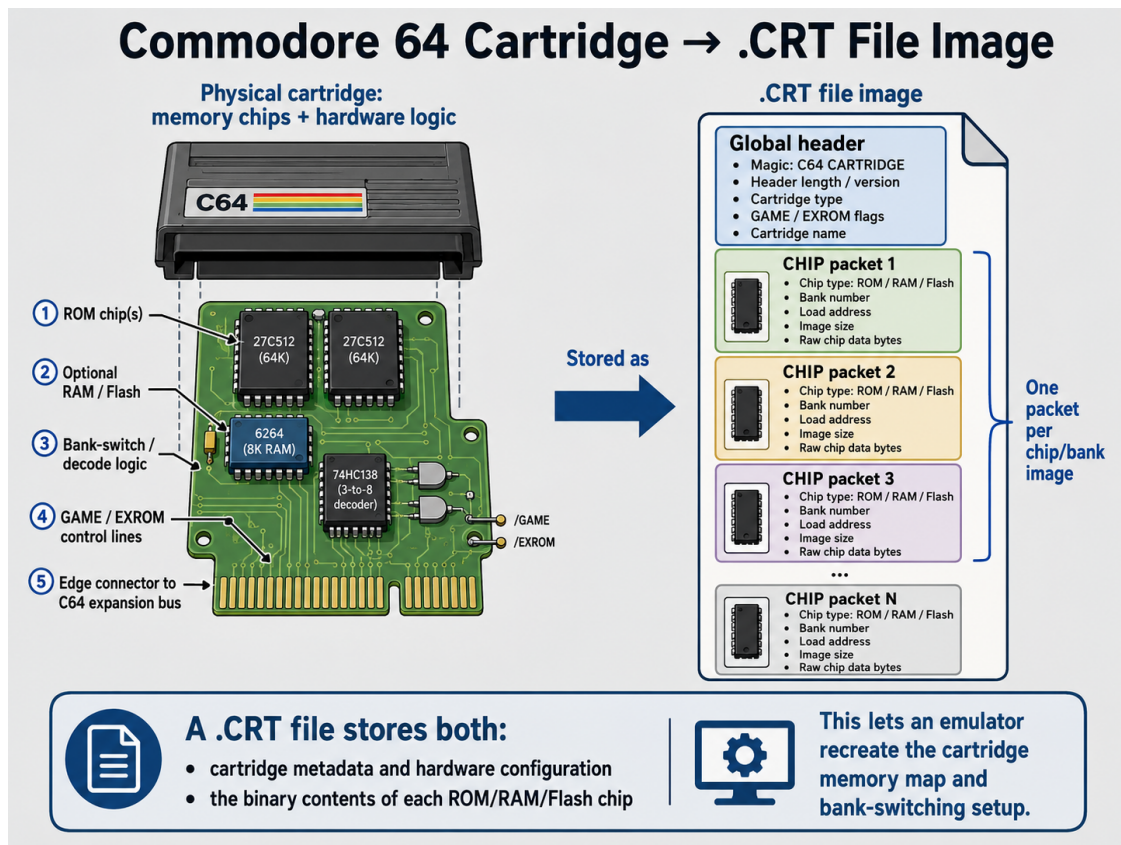


Figure 4.6: Illustrating the relation between Commodore 64 Cartridges and the .CRT file image.

4.5 Display and Audio setup

CCS64 normally starts in windowed mode. However, using the "Video..." sub-menu it is possible to select other display modes, that includes full-screen resolutions. Note that full-screen modes are handled by DirectX for rendering, which can interfere with the Windows workbench due to the change in display resolution. There are several graphical filters to choose from - spanning from simple filters up to advanced CRT TV emulation. The rendering of both the windowed and full screen modes can be rotated if that is required by the emulated program or the physical display used.

By using the "Sound..." sub-menu it is possible to configure the quality of the audio sample playback. It is also possible to select a different version (8580) of the SID chip than the default (6581) and configure the filter parameters.

4.6 Speed Control and Pause mode

CCS64 normally runs at the regular speed of the Commodore 64. However, it is possible to control the speed with which the processor and the rest of the Commodore 64 system is emulated. By the means of the "Options..." sub-menu or the corresponding fast-keys it

is possible to speed up (e.g. 5x, or by an arbitrary factor up to maximum) or slow down (arbitrary slow-motion from 100 down to 1 %) the emulation.

The emulation speed can also be completely paused anytime by pressing Alt+P on the keyboard or by using the appropriate GUI buttons, sliders and menus. This feature behaves similar to what certain C64 cartridges called as a freeze function. In pause mode, you can press [P] (or left cursor key) or [N] (or right cursor key) to step a single frame (or one whole second of frames if holding down shift key simultaneously) backwards or forwards in emulation history. The pause mode can be exited by pressing Alt+P again.

4.6.1 Speed-o-meter

To enable an overlay display, continuously showing the current emulation speed (as percentage of normal) and graphics update frequency (as frames per second), select the corresponding option in the "Special..." sub-menu.

4.7 Saving states

By opening the "State..." sub-menu or using the corresponding fast-keys, it is possible to store a complete snap-shot (i.e., a freeze function) of the whole C64 emulation state, that can be later loaded in order to restore the exact state. The states can either be saved to the computer's memory or to the file system. Note that the files can be relatively large.

4.8 Screenshots and Audio/Video Recording

By opening the "Video..." sub-menu and selecting the "Snapshot..." option, it is possible to make a screen snap-shot. This feature saves an image of the whole current C64 screen (maximum 504x312 pixels for PAL, 520x263 for NTSC) in 16-, 256- or 16777216-bit colors as a .BMP file.

By opening the "Video..." sub-menu and selecting the "Video Sampling" option, it is possible to start a recording of the emulation session as an .AVI file (including both audio and video).

By opening the "Sound..." sub-menu and selecting the "Sound Sampling" option it is possible to start a recording of the emulation session as a .WAV file (including only audio).

Fast-Key	Description
Alt+8	Menu for 1541 device 8. Press twice for Status Menu.
Alt+9	Menu for 1541 device 9. Press twice for Status Menu.
Alt+0	Menu for 1541 device 10. Press twice for Status Menu.
Alt++	Menu for 1541 device 11. Press twice for Status Menu.
Alt+1	Menu for Tape device 1. Press twice for Status Menu.
Shift+Alt+8	Status Menu for 1541 device 8.
Shift+Alt+9	Status Menu for 1541 device 9.
Shift+Alt+0	Status Menu for 1541 device 10.
Shift+Alt++	Status Menu for 1541 device 11.
Shift+Alt+1	Status Menu for Tape device 1.
Alt+R	Re-set (Soft).
Shift+Alt+R	Re-set (Hard).
Alt+O	Options Menu.
Alt+V	Video Menu.
Alt+I	Input Menu.
Alt+M	MC (Machine Code) Monitor.
Alt+E	Special Menu.
Alt+S	Sound Menu.
Alt+C	Cartridge Menu.
Alt+T	State Menu.
Alt+F	Cartridge Freeze button.
Alt+P	Pause on/off.
Alt+B	Backwards playback. Uses the current accumulated emulation history to apparently run the emulator backwards in full C64 speed.
Shift+Alt+P	PAL TV palette filter emulation on/off.
Alt+Q/X	Quit.
Alt+F1	Save 16-bit color screen-shot.
Shift+Alt+F1	Save PAL/RGB color screen-shot.
Alt+F2	Save 256-bit color screen-shot.
Alt+F3	Toggle Speed (100 - 500 %).
Alt+F4	Quit.
Alt+F5	Insert next .D64/.G64 disk image into device 8.
Alt+F6	Insert previous .D64/.G64 disk image into device 8.
Alt+F7	Select next input device/mode in C64 game port 1.
Alt+F8	Select next input device/mode in C64 game port 2.
Alt+F10	Exchange (switch) input device/mode settings between C64 game port 1 and 2.
Alt+F11	Loads a frozen emulator state from selectable file-name.
Alt+F12	Saves a frozen emulator state to selectable file-name.
Shift+Alt+F2	Screen-shot current text menu display.
Shift+Alt+F3	Toggles the warp speed (maximum emulation speed) option.
Alt+K	Insert special C64 key-stroke.
Shift+Alt+S	Toggles the recording of sound to a WAV file.
Shift+Alt+V	Toggles the recording of video+sound to an AVI file.

Table 4.1: Overview of the fast-keys available during the emulation session.

Chapter 5

Supported Hardware and Emulation

The Commodore 64 is not a single processor running software, but rather a collection of specialized hardware components that cooperate to form a complete computer system. The central processor, graphics and sound chips, interface controllers, memory system, storage devices, and expansion hardware each contribute to the overall behavior of the machine. Many software programs interact directly with these hardware components and frequently depend upon their precise timing, undocumented characteristics, or electrical behavior.

CCS64 therefore emulates the Commodore 64 as an interconnected hardware system rather than merely executing 6510 instructions. Each major device is reproduced with the objective of preserving both its individual functionality and its interaction with the remaining hardware. Depending upon the subsystem, this may include clock-cycle timing, memory arbitration, bus communication, interrupts, peripheral protocols, analog video characteristics, or mechanical storage behavior.

The following sections provide an overview of the principal hardware components emulated by CCS64 and describe the techniques used to reproduce their behavior accurately. Rather than documenting every hardware feature in detail, the intention is to explain the role of each device within the Commodore 64 architecture and the aspects that are particularly important for achieving compatibility, historical preservation, and faithful emulation.

5.1 CPU

The Commodore 64 is based around the MOS Technology 6510 microprocessor, which is closely related to the well-known 6502 CPU family but extended with additional on-chip I/O functionality used for memory banking and system control. In the Commodore 64, the processor does not operate in isolation; instead, it continuously interacts with the VIC-II graphics chip, the CIA interface controllers, cartridge hardware, and the external bus shared with peripherals such as the 1541 disk drive. Correct emulation therefore requires not only instruction compatibility, but also accurate timing and synchronization between multiple hardware components.

CCS64 emulates the 6510 processor at cycle level, meaning that individual clock cycles, memory accesses, interrupts, and bus interactions are reproduced in the same sequence as on the original hardware. This includes many undocumented or hardware-specific side

effects relied upon by advanced software, especially demos, fast loaders, copy protections, and timing-sensitive games. The emulator also reproduces interactions between the CPU and the VIC-II during bad lines, sprite DMA activity, and raster interrupts, where memory access timing becomes critical.

The processor emulation has been developed with a strong focus on hardware accuracy and compatibility rather than simplified high-level execution. As a result, CCS64 is capable of running software that depends upon exact instruction timing, undocumented opcodes, self-modifying code, cycle-stable raster effects, and low-level synchronization with external hardware. This level of emulation accuracy is particularly important within the Commodore demo scene, where software often intentionally exploits undocumented hardware behavior in order to achieve advanced visual or audio effects.

5.2 VIC-II

The VIC-II graphics chip is one of the most distinctive and technically advanced components of the Commodore 64 system. Besides generating the video output displayed on the screen, the VIC-II also controls raster timing, sprite rendering, border generation, scrolling, light-pen input, and direct memory access (DMA) operations that temporarily suspend the CPU. Many visual effects used by games and demos depend not only on the documented functionality of the VIC-II, but also on precise timing relationships between raster positions, memory fetch cycles, and internal hardware state changes.

CCS64 emulates the VIC-II at pixel and cycle level, reproducing the internal video generation process on a clock-by-clock basis. The emulator models raster progression, sprite evaluation, bad lines, border behavior, sprite collisions, memory fetch timing, and other low-level operations used by timing-sensitive software. Multiple VIC-II variants are supported, including both PAL and NTSC models with their differing raster timings, clock frequencies, and screen geometries. Depending on the hardware model being emulated, the VIC-II therefore operates with either 63, 64, or 65 clock cycles per raster line and with either 312, 263, or 262 raster lines per frame. These timing differences are essential for compatibility with software that relies upon cycle-stable raster synchronization or hardware-specific display behavior. Particular attention has been given to undocumented and hardware-dependent behavior discovered through hardware testing and reverse engineering within the Commodore community. This includes many effects that rely upon exact raster synchronization or intentionally exploit internal VIC-II timing characteristics.

The graphical output generated by the VIC-II is further processed by CCS64 in order to reproduce characteristics of historical CRT monitors and televisions. PAL and NTSC display behavior, scanlines, color blending, analog filtering, and RF-related artifacts can therefore be approximated in software to better reflect the appearance of the original hardware. The purpose of this approach is not only graphical authenticity, but also software compatibility, since many Commodore 64 productions were designed while considering the visual properties of analog CRT displays.

5.3 SID

The SID (Sound Interface Device) chip is responsible for audio generation in the Commodore 64 and is widely regarded as one of the most characteristic sound chips of the home computer era. Besides providing three independent audio voices with programmable waveforms, filters, modulation, and envelope control, the SID also contains numerous analog characteristics and undocumented behaviors that contribute significantly to the unique sound associated with the Commodore 64. Different hardware revisions, particularly the MOS 6581 and MOS 8580 models, also exhibit noticeable electrical and acoustic differences.

CCS64 emulates both the digital and analog aspects of the SID architecture. The digital core, including waveform generation, ADSR envelopes, oscillators, synchronization, ring modulation, and timing behavior, is emulated at cycle level in synchronization with the rest of the emulated hardware. Particular care has been taken to reproduce timing-sensitive behavior and side effects relied upon by music players, demos, and advanced sound routines. The emulator synthesizes audio samples directly from the internal SID state rather than relying on pre-recorded approximations.

The analog characteristics of the SID, especially the filter circuitry and amplifier stages, are inherently more difficult to reproduce exactly because physical SID chips vary significantly between manufacturing revisions and even between individual units. CCS64 therefore approximates these analog properties using mathematical models and configurable parameters intended to reproduce the overall acoustic behavior of the original hardware. The purpose is not only musical authenticity, but also compatibility with software that intentionally exploits analog SID behavior, waveform leakage, filter resonance, or timing-related audio effects commonly used within the Commodore music and demo scenes.

5.4 CIA

The Commodore 64 contains two MOS 6526 Complex Interface Adapter (CIA) chips that provide much of the system's input/output functionality and timing control. The CIA chips are responsible for keyboard scanning, joystick input, timers, interrupt generation, serial bus communication, TOD (Time-of-Day) clocks, and interaction with external peripherals. Although less visible than the VIC-II or SID, the CIA devices play a central role in overall system behavior and are heavily relied upon by operating system routines, games, fast loaders, and synchronization code.

CCS64 emulates the CIA chips at cycle level, including timers, interrupt logic, serial communication behavior, and internal register interactions. Accurate timing is particularly important because many Commodore 64 programs synchronize operations against CIA timers or depend upon exact interrupt behavior. The emulator therefore reproduces not only documented CIA functionality, but also numerous hardware-specific timing details and edge cases observed on real hardware.

The interaction between the CIA chips and the rest of the Commodore 64 system is especially important during disk communication, raster synchronization, keyboard scanning, and real-time software control. Advanced software frequently combines VIC-II raster interrupts with CIA timer interrupts in order to achieve stable timing behavior across video frames.

CCS64 therefore maintains close synchronization between the CIA devices, CPU execution, video timing, and external peripherals such as the emulated 1541 disk drive. This level of coordination improves compatibility with timing-sensitive software and contributes to more faithful reproduction of original hardware behavior.

5.5 1541 Disk Drive

The Commodore 1541 floppy disk drive is not merely a storage device, but a complete computer system containing its own MOS 6502 processor, RAM, ROM firmware, VIA interface controllers, and disk read/write electronics. Communication between the Commodore 64 and the 1541 takes place over the serial IEC bus, while many commercial games and demo productions additionally rely upon custom fast loaders, copy protection schemes, or direct synchronization between the computer and the disk drive. As a result, accurate emulation of the 1541 requires substantially more than simply reading files from disk images.

CCS64 emulates the 1541 as an independently executing hardware subsystem synchronized in real time with the Commodore 64 itself. The internal 6502 processor, VIA 6522 interface controllers, serial bus timing, head stepping mechanics, disk rotation behavior, and GCR (Group Code Recording) encoding are all reproduced as part of the emulation process. Particular attention has been given to low-level disk behavior such as synchronization marks, varying track densities, half-track positioning, weak bits, and custom sector layouts commonly used by copy-protected software and advanced fast loaders.

The disk-drive emulation operates at cycle level and continuously interacts with the emulated Commodore 64 through shared timing and bus communication. This synchronization is important because many software loaders depend upon exact drive timing, precise interrupt behavior, or parallel execution between code running simultaneously in both the C64 and the 1541 drive CPU. CCS64 therefore emulates not only the visible file contents of floppy disks, but also much of the underlying physical and electrical behavior of the original drive hardware.

In addition to standard D64 disk images, CCS64 supports lower-level G64/G41 formats capable of preserving detailed disk structure and timing information. Multiple disk drives can be connected simultaneously, and optional hardware extensions such as parallel cables and drive RAM expansions are also supported.

CCS64 also provides a “Maximum 1541 Speed” mode that accelerates disk loading and saving operations by temporarily running the emulation faster than real-time during active drive access. Unlike simplified fast-load techniques that bypass parts of the original loading process, CCS64 continues to execute the Commodore 64 and 1541 systems normally and cycle-accurately, including all communication, synchronization, and loader code. The effect is therefore not a reduction in emulation accuracy, but rather a compression of the waiting time experienced by the user. Software that could require minutes to load on original hardware can consequently start within seconds on modern computers while still behaving as if the complete loading process had occurred on a real Commodore 64 and 1541 system.

5.6 Tape

The Commodore 64 supports program loading and data storage through the Commodore C2N cassette tape system, where digital information is represented as sequences of analog pulses recorded on ordinary audio tape. Compared to floppy-disk based systems, cassette storage is considerably slower and more sensitive to timing variations, signal quality, and loader-specific encoding techniques. Nevertheless, tape-based software distribution was extremely common during the early years of the Commodore 64 and remains historically important for software preservation.

CCS64 emulates the cassette subsystem at signal level, including tape motor control, play-button sensing, pulse generation, and timing-sensitive tape reading behavior. The emulator supports both high-level tape archive formats such as T64 and low-level raw-tape formats such as TAP. While T64 files primarily preserve the logical program contents stored on tape, TAP files preserve the original pulse timing information itself, allowing reproduction of custom loaders, turbo-tape systems, synchronization schemes, and many forms of non-standard tape encoding used by commercial software and hobby productions. See Section 4.3 for loading tape images.

The tape emulation is designed to preserve not only the data stored on cassette media, but also much of the operational behavior of the original hardware system. Loading therefore (optionally) occurs through the normal Commodore tape routines and timing mechanisms rather than by directly injecting program data into memory. This approach improves compatibility with software that depends upon exact tape timing or multi-stage loading sequences. CCS64 can additionally reproduce the characteristic mechanical and acoustic behavior of the cassette system, including optional playback of tape-drive noise during loading operations, further contributing to the authenticity of the emulation experience.

The preservation of raw-tape formats is particularly important because many historical cassette releases contain timing characteristics, custom loaders, or analog signal properties that cannot be fully represented by simplified archive formats alone. CCS64 therefore treats cassette support not merely as a convenience feature, but also as part of a broader effort toward accurate preservation of original Commodore 64 software distribution media and hardware behavior.

5.7 Cartridges

The Commodore 64 expansion port provides direct electrical access to the system address bus, data bus, control lines, and memory-management signals, allowing external cartridges to extend or modify the behavior of the computer at hardware level. Cartridge hardware was historically used for a wide variety of purposes, including games, fast loaders, programming tools, RAM expansion, freezer systems, operating-system extensions, and custom hardware accelerators. Unlike ordinary software loaded from disk or tape, cartridge hardware can actively participate in the memory decoding process of the Commodore 64 and dynamically alter the visible memory configuration of the machine while it is running.

A central aspect of cartridge emulation in CCS64 is therefore the reproduction of the Commodore 64 memory-management architecture itself. The standard memory banking of

the C64 is controlled internally through the 6510 processor I/O register, whose control bits determine whether RAM, BASIC ROM, KERNAL ROM, character ROM, or I/O regions are visible within different address ranges. However, cartridge hardware additionally influences the memory configuration through the two expansion-port control signals named EXROM and GAME. These signals interact with the internal MMU logic and effectively select between several fundamentally different system-wide memory configurations.

The combination of the 6510 banking register together with the EXROM and GAME signals creates multiple overlapping memory-mapping modes that determine which devices and memory regions are visible to the CPU at any given moment. Particularly important is the Ultimax mode, originally derived from the Commodore MAX Machine architecture, where large portions of the normal RAM map disappear and cartridge ROM becomes directly mapped into the processor address space. In this configuration, many ordinary assumptions about the standard Commodore 64 memory layout no longer apply. Advanced cartridges frequently manipulate these signals dynamically during execution in order to switch ROM banks, intercept memory accesses, activate freezer hardware, or expose custom registers and peripherals.

CCS64 emulates the cartridge subsystem by reproducing both the electrical memory-selection behavior and the timing interactions between cartridge hardware and the rest of the emulated system. Cartridge hardware is therefore not treated merely as static ROM contents, but as active hardware devices capable of controlling memory visibility and interacting directly with CPU execution. This approach improves compatibility with cartridges that depend upon dynamic bank switching, custom address decoding, DMA-like behavior, freezer logic, or precise synchronization with the Commodore 64 bus system.

The emulator supports the CRT cartridge image format, which preserves not only cartridge ROM contents but also metadata describing hardware type and banking behavior. Through this mechanism, CCS64 is able to emulate a wide range of historically important cartridge systems while maintaining compatibility with the underlying Commodore 64 memory-management architecture. This includes the normal (plain ROM/RAM without I/O) type of cartridge and the following specific cartridge types: Action Replay, KCS Power Cartridge, Final Cartridge I & III, Simons Basic, Ocean Games, Fun Play, Super Games, Atomic Power, Epyx Fastload, Westermann, Rex, C64 Game System, WarpSpeed, Dinamic, Zaxxon, Expert, Magic Desk, Super Snapshot 5, Comal 80, Magic Formel, Ross, Structured BASIC, Dela-EP64, REX 256, Mikro Assembler, Final Cartridge Plus, Action Replay 4, StarDOS, and EasyFlash.

5.8 REU

The Commodore 1764 RAM Expansion Unit (REU) provides additional external memory connected through the Commodore 64 expansion port. Unlike ordinary memory expansions directly mapped into the CPU address space, the REU operates largely as an independent DMA-based transfer device capable of rapidly copying data between the Commodore 64 memory and external RAM without continuous CPU involvement. REU hardware was historically used by advanced applications, operating systems, utilities, and some demo productions requiring larger memory capacity or high-speed memory transfers.

CCS64 emulates the REU at hardware level, including DMA transfer behavior, address-control logic, transfer modes, and timing interactions with the Commodore 64 bus system. The emulation supports configurable REU memory sizes ranging from the original 1764 configuration (128 KB) up to larger extended capacities (16 MB). Particular attention has been given to synchronization between the REU and the main system timing, since DMA operations temporarily take control of the memory bus and therefore directly influence CPU execution timing and overall machine behavior.

The REU emulation can additionally operate simultaneously together with cartridge hardware and other expansion devices, allowing reproduction of more advanced hardware configurations used by certain software environments and hardware expansions.

5.9 Printers

CCS64 provides high-level emulation of Commodore printer devices connected through the IEC serial bus using device numbers 4 to 7. The printer support is implemented by intercepting relevant Commodore KERNAL routines, including OPEN, CLOSE, CHRIN, CHROUT, CHKIN, and CHKOUT, thereby allowing standard BASIC and KERNAL-based software to produce printed output without requiring full low-level printer hardware emulation.

The printer emulation does not simulate the electrical or protocol-level behavior of physical IEC printer devices. Instead, printed output is redirected internally by the emulator and typically stored as ordinary text files on the host computer system. This approach provides practical compatibility with many applications and BASIC programs while avoiding the complexity of cycle-exact emulation of specific Commodore printer hardware implementations.

5.10 Input devices

The Commodore 64 supports a variety of input devices connected through the keyboard matrix, control ports, and user port interfaces. Besides the standard keyboard, common input devices include digital joysticks, paddles, mice, light pens, and various custom controllers used by games and specialized software. Many Commodore 64 programs interact directly with the underlying hardware registers of the CIA interface controllers rather than relying upon operating-system routines, making accurate emulation of input timing and port behavior important for compatibility.

CCS64 emulates the Commodore 64 keyboard at matrix level, including support for all standard keys as well as special keys such as RESTORE. Keyboard input from the host system is translated into corresponding C64 key matrix states, allowing both ordinary typing and low-level keyboard scanning routines to operate correctly. The emulator additionally includes functionality for inserting native C64 key combinations and special character sequences that may not exist directly on modern PC keyboards.

Joystick support is available both through host-system game controllers and through keyboard emulation. The emulator supports the two standard Commodore joystick ports, including automatic joystick-port detection for software that dynamically switches between

ports 1 and 2. Multiple fire buttons can optionally be mapped to host input devices even though most original Commodore joysticks only provided a single button. CCS64 also supports additional control-port devices such as the Commodore 1351 mouse and analog paddles.

The interaction between input devices and the Commodore 64 hardware is emulated in coordination with the CIA chips and system timing, allowing compatibility with software that performs direct hardware polling, raster-synchronized input reading, or custom joystick and keyboard scanning routines.

5.11 Displays

The graphical output generated by the Commodore 64 was originally intended for display on analog CRT monitors and television sets operating according to either the PAL or NTSC video standards. The visual appearance of Commodore 64 software therefore depended not only on the digital output generated by the VIC-II graphics chip, but also on numerous analog properties introduced by the video encoding, transmission, and display hardware. These effects include scanline behavior, phosphor persistence, color bleeding, luminance and chrominance interference, RF transmission noise, analog filtering, and pixel interactions caused by the physical structure of CRT displays. See Figure 5.1 for an graphical illustration of how the VIC-II produces graphics that are transformed and transmitted as a RF-signal for display on a CRT Color TV.

CCS64 emulates the display system as a multi-stage process extending beyond the internal digital VIC-II graphics generation itself. The emulator can reproduce PAL and NTSC video behavior at both line and pixel level, including color artifacts caused by phase relationships, chroma/luma interference, and analog decoding characteristics of historical television systems. This includes effects where neighboring pixels influence perceived color output differently depending on horizontal position, raster alignment, or video standard. Such artifacts were frequently exploited by games and demos in order to produce additional apparent colors, smoother gradients, or visually blended graphics beyond the nominal limitations of the digital VIC-II palette.

The display emulation additionally models several physical properties associated with CRT technology itself. These include phosphor decay and persistence, scanline intensity variation, shadow-mask and aperture-grille structures, analog blur characteristics, and temporal blending between consecutive frames. CCS64 can furthermore approximate electrical and transmission-related imperfections such as RF modulation noise, signal interference, and analog instability commonly associated with home television connections used with original Commodore 64 systems.

The purpose of these display simulations is not merely aesthetic reproduction, but also historical and technical accuracy. Many Commodore 64 software productions were designed while taking the behavior of analog PAL or NTSC displays into account, meaning that the appearance on modern digital displays can differ substantially from the visual result originally intended by software developers and artists. CCS64 therefore treats display emulation as an integral part of reproducing the overall behavior and presentation characteristics of the original Commodore 64 platform.

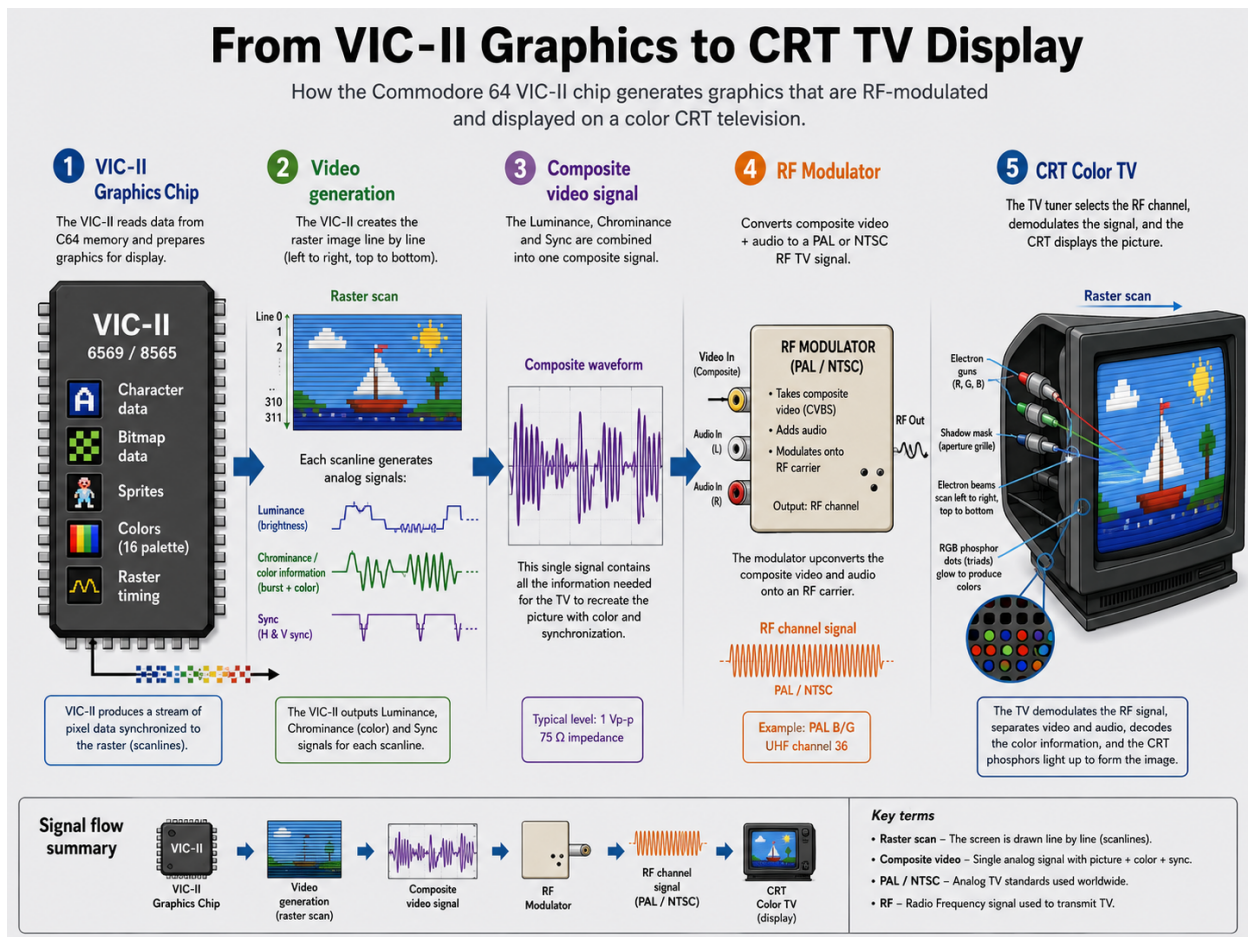


Figure 5.1: An illustration of how the VIC-II graphics chip render graphics that are transformed into a TV RF-modulated signal (PAL or NTSC) that is then displayed by a CRT Color TV.

Chapter 6

Advanced Features

6.1 Rewind/Replay

Unlike conventional save states, which capture the state of the emulated machine at a single instant in time, the rewind/replay system continuously records the evolution of the Commodore 64 while it is running. This accumulated execution history allows the user to move both forwards and backwards through previously executed emulation, making it possible to revisit earlier moments without interrupting normal operation or manually creating save states in advance.

The replay history records the complete state of the emulated Commodore 64 system, including memory contents, processor state, hardware devices, and attached peripherals. As emulation progresses, older history entries are gradually discarded to make room for new ones, resulting in a continuously moving time window whose duration depends upon the amount of memory allocated to the replay buffer. The default duration is 5 minutes, and is configurable up to 60 minutes. Within this recorded interval, the emulator can reproduce the previous execution exactly as it originally occurred.

See Figure 6.1 for an image of the slider control that provides an easy and powerful means to rewind the emulation to any point within the time window. The rewind/replay functionality is useful in many different situations. Games can be resumed from moments before an unsuccessful action, software demonstrations can be replayed repeatedly, and developers can revisit earlier execution states when investigating programming errors or unexpected hardware behavior. Combined with the machine-code monitor, the replay system provides a powerful debugging environment where execution can be examined both before and after an observed event. Rather than forcing the user to predict when a problem may occur, CCS64 makes it possible to investigate the sequence of events that actually led to it.

Unlike video recording, replay preserves the complete internal state of the emulated hardware rather than simply storing screen images and sound. Consequently, replayed execution remains fully interactive. The user may pause the replay at any point, inspect or mod-

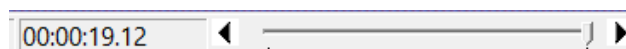


Figure 6.1: Slider control for fast and flexible rewinding and replay of the emulation history.

ify memory and registers, enter the machine-code monitor, or continue execution along a completely different path. The replay system therefore functions not merely as a recording mechanism, but as an extension of the emulation itself, providing a practical means of exploring, analysing, and interacting with Commodore 64 software across time.

6.2 Reverse Execution

Reverse execution builds upon the rewind/replay system described in the previous section by providing continuous backwards playback of the recorded emulation history. Instead of manually stepping through previously executed frames, CCS64 automatically rewinds the replay history while continuously updating both the display and the audio. The result is the appearance of the Commodore 64 running backwards in real time, effectively allowing the emulation to operate at approximately -100% speed.

This effect is achieved without executing the emulated Commodore 64 hardware in reverse. Rather, the emulator successively restores previously recorded machine states from the replay history, displays each frame in reverse chronological order, and plays the corresponding audio samples backwards. The resulting playback provides a smooth and intuitive means of revisiting earlier events while preserving the exact visual and auditory output that was originally produced.

Reverse execution is particularly useful when investigating events that have just occurred during gameplay or program execution. Rather than repeatedly moving a rewind slider or stepping backwards frame by frame, the user can simply watch the emulation history unfold in reverse until the desired point is reached. At any moment, the reverse playback may be stopped and normal forward emulation resumed from the selected point in time, allowing interactive exploration without the need to predict in advance when an interesting event would occur.

6.3 Kaillera Multi-play

CCS64 supports online multiplayer through the Kaillera networking system, allowing several users to participate in the same Commodore 64 session across a local network or the Internet. Each participant runs an identical copy of the emulated machine and software, while player input is synchronized between the connected systems. Because every emulator executes the same sequence of operations, the original game logic, timing, and hardware behavior are preserved without modifying the software being played.

From the main menu, enter the sub-menu of Device 1 or 1541 Device 8-11. Enter the directory that contains the .T64, .D64, or .PRG files that you want to be able to use for multi-play over Internet. Use the [Cursor Up/Down] keys to high-light the "." directory, then press the [F3] key. Using the Kaillera client, you can now connect to a server, and from there, either create a new game or join in to an existing one. Remember to not start your created game until enough people have joined. If you join an existing game, the game will start as soon as the game creator has started. To be able to join an existing game, you have to have started the multi-play menu from a directory which contains exactly the same file as the one that the game creator has.

During the active multi-play session, no menus can be reached and only a limited subset of fast-keys are supported. Therefore, remember to set correct settings for input devices etcetera, before entering the multi-play menu. The multi-play session is ended by pressing either the [F9] or [F10] keys.

The following fast-keys work and are distributed to all players:

Alt+F5 Insert next .D64/.G64 disk image into device 8.

Alt+F6 Insert previous .D64/.G64 disk image into device 8.

The following fast-keys work and are only local to the current player:

Alt+F7 Select next input device/mode in C64 game port 1.

Alt+F8 Select next input device/mode in C64 game port 2.

Alt+F10 Exchange (switch) input device/mode settings between C64 game port 1 and 2.

Chapter 7

Machine-Code Monitor

The CCS64 machine-code monitor is an integrated development and debugging environment for the Commodore 64. It can be used to write and modify 6510 machine code, inspect and alter memory, analyze program execution, trace hardware behavior, and diagnose software problems. The monitor is equally useful for developing new software, reverse engineering existing programs, and learning how the Commodore 64 operates at the hardware level. Once familiar with the basic commands, it becomes a powerful tool for both beginners exploring assembly language and experienced programmers debugging complex applications. The following sections describe how to enter the monitor, navigate its interface, and use its commands effectively.

7.1 Starting

The native machine-code monitor can be invoked either from the Menu or by using the corresponding fast-key (Alt+M). As the monitor is native it doesn't affect the current emulated program in any way, all memory is untouched - unless you do a change yourself, that is. The monitor is exited with the Escape key.

The user interface is designed as a full screen editor, which means that you can navigate to previously executed commands and their output using the cursor keys. The editor's contents also scrolls upwards and downwards when the cursor tries to move out of the vertical boundaries, and automatically continues the execution of previously executed commands. You can swap to one of the 8 different session screens by pressing the Alt key together with one of the keys 1 to 8:

Alt+1	Session 1
Alt+2	Session 2
...	
Alt+8	Session 8

You can always get up the internal help and various status pages by pressing the function keys F1 to F4:

Command Syntax	Description
A <xxxx> <mnemonic>	Assemble. Example: A 1000 LDA #\$09
BANK [<xx>]	Bank change 0-7 for different C64 MMU maps, 8-11 for the 1541 device.
B	Show break-points.
BA <xxxx>[-<yyyy>] [RWX] [<xx>[-<yy>]]	Adds a break-point at the specified address [xxxx] or within the specified address range [xxxx-yyy]. It is triggered when the address [xxxx] or anywhere within the address range [xxx - yyy] is Read from, Written to, or Executed. Default is any access triggers the break-point. Example: BA D012 W F7-FF - This will only trigger the break-point whenever a program writes a value between F7 to FF to address D012.
BR <xx>	Remove the specified break-point. Example: BR 0 - This will remove the first break-point.
C <xxxx> <yyyy> <zzzz>	Compare memory.
CD [<path>]	Changes directory to the specified path or prints the current working directory if no path is specified. MS-DOS path-names can be used and Long path-names can be used (if the Long path-name is enclosed in " signs). The / sign can be used instead of the \ sign.
D <xxxx> [<yyyy>]	Dis-assemble.
DDD [<xx>]	Disk-drive debugging.
DIR [<path>]	Displays the contents of the directory path specified or prints the contents of the current working directory if no path is specified. MS-DOS path-names can be used and Long path-names can be used (if the Long path-name is enclosed in " signs). The / sign can be used instead of the \sign. Directory listings can be interrupted by pressing <ESC>.
F <xxxx> <yyyy> <xx> ...	Fill memory with Hex values or ASCII strings. Example: F 1000 2000 "HELLO" 0D 0A

Table 7.1: Commands (A-F).

F1	Help
F2	VIC-II Status
F3	CIA1 Status
F4	CIA2 Status

7.2 Command Overview

The commands that are supported are briefly presented in Tables 7.1, 7.2 and 7.3.

Some of the commands that can potentially produce several lines of output, like the M command, can be manually continued by scrolling up/down with the [Cursor Up/Down] keys to the screen limits, or using the [Page Up/Down] keys. To get a free line to enter a new command, just press the [Enter] key.

Command Syntax	Description
HD <xxxx> <yyyy> <string>	Hunt dis-assembly output textually for patterns. Wild-cards are expressed with *. Example: HD A000 B000 "LDA*\$*Y"
H <xxxx> <yyyy> <xx> ...	Hunt memory for Hex values, ASCII and wilds (XX). Example: H 1000 2000 10 XX "LEVEL"
I <xxxx> [<yyyy>]	Interpret memory as ASCII.
L "<filename>" <xxxx>	Loads memory from binary file (not .prg). Example: L "Hack" 0800 1000
LP "<filename>" <xxxx>	Loads memory from binary program file (.prg). Example: LP "Hack.prg"
M <xxxx> [<yyyy>]	Shows memory as Hex values.
P <xx>	Printer re-direct. Re-directs output of next D or M commands to the specified printer device (4-7).
Q <xxxx>	Quick-traces until the PC reaches a certain address.
R <reg> <xx..> ...	Register. Changes the contents of the named CPU register. Example: R PC FCE2 A 0 - This will set the PC to FCE2 and A to 0.

Table 7.2: Commands (H-R).

Command Syntax	Description
S "<filename>" <xxxx>	Saves memory to binary file (not .prg). Example: S "Picture" 2000 3F40
SP "<filename>" <xxxx>	Saves memory to binary program file (.prg). Example: SP "Demo.prg" 2000 3F40
T <xxxx> <yyyy> <zzzz>	Transfer memory. Copies the contents of a memory region into another.
VD	VIC-II debugging. Enables interactive debugging forwards and backwards in both x-axis and y-axis using the cursor keys , with extra focus on the current VIC-II status besides the 6510 status.
W [<xxxx>]	Traces forwards with single-step.
WB [<xxxx>]	Traces backwards with single-step.
X	Exits monitor.
: <xxxx> <xx> ...	Changes memory from Hex values.
, <xxxx> <xx> ...	Changes memory from dis-assemble Hex values.
* [<xx>]	Bank change 0-7 for different C64 MMU maps, 8-11 for the 1541 device.
? <expression>	Evaluate a numeric expression and display the result in decimal and hexa-decimal. The syntax of the expression is as follows: <i>expr ::= hexnr '\$hexnr' '#'decnr '-'expr expr '+expr expr '-'expr expr '*expr expr '/'expr expr '%expr '('expr ' Example: ?23+(56%7+55*(#23-7/3))</i>

Table 7.3: Commands (S-?).

```

PC      CV      SR      AC      XR      VR      SP      IR0      NH1      BANK
ESD1 0000 32 00 00 0A F3 EA31 FE47 3:07

A1000 AD 04 DC LDA $DC04
A1003 8D 20 D0 STA $D020
A1006 4C 00 10 JMP $1000
A1009

D 1000 1009
:1000 AD 04 DC LDA $DC04
:1003 8D 20 D0 STA $D020
:1006 4C 00 10 JMP $1000

M 1000 1010
:1000 AD 04 DC 8D 20 D0 4C 00 00 00 00 00 00 00 00
:1008 10 FF FF FF FF FF FF FF FF 00 00 00 00 00 00
M 1000 1010
:1000 AD 05 DC 8D 20 D0 4C 00 00 00 00 00 00 00 00
:1008 10 FF FF FF FF FF FF FF FF 00 00 00 00 00 00

D 1000 1009
:1000 AD 05 DC LDA $DC05
:1003 8D 20 D0 STA $D020
:1006 4C 00 10 JMP $1000

```

Figure 7.1: Example showing the Assemble (A) command, followed by the Disassemble (D) command and the Memory (M) command.

7.3 Guides

7.3.1 Coding

To start entering an assembly program, move to an empty line and enter "A <address> <mnemonic>" and then press [Enter] to convert the mnemonic into machine code that is stored at the subsequent bytes in memory at the specified address. For example, if you enter "A 1000 LDA DC04", this will change to "A 1000 AD 04 DC LDA \$DC04" - indicating that the mnemonic "LDA DC04" has been successfully interpreted and converted into the machine code sequence of AD 04 DC. Note that hexadecimal values are expected as default, and thus it is not mandatory to specify the \$ sign.

To interpret machine code into assembly mnemonics, move to an empty line and enter "D <address>" and then press [Enter] to start the disassemble command. See Figure 7.1 for an example showing a session starting at the top with the A command to enter a short assembly program consisting of three lines. Thereafter, the D command is used to disassemble the machine code - thus checking that the program was correctly stored into the memory.

7.3.2 Memory

To view the contents of a memory interval, move to an empty line and enter "M <address>" and then press [Enter] to show the subsequent bytes at the specified address. The command will present the contents as ":<address> <byte> ... <byte> <ascii> ... <ascii>" for eight bytes at a time. To change the memory contents, simply navigate to the corresponding two-digit byte displayed - then change it and press [Enter]. See Figure 7.1 for an example showing a session that uses the M command to display the memory contents of a previously assembled machine code program, two times after each other. Thereafter, the cursor is moved to the position of the second byte of the machine code program (corresponding to the address \$1001) and the byte is changed to the value 05. Note that the M command also displays the eight bytes as ASCII characters. If you are only interested in displaying the

```

PC      CV      SR      AC      XR      YR      SP      IRQ      NMI      BANK
F03C  3EAA  36 00  21 0C  F9  0000  0000  3:05

BA D01F R
X
M
#BREAK #0 ON READING $00 AT $D01F
F0D8 85 5B      STA $5B      ***
F0DA 60          RTS          *****
F0DB 20 38 F0    JSR $F038    *****
F038 A5 5B      LDA $5B      ***
F03A 29 01      AND $201      **
F03C F0 03      BEQ $F041    0000
F03A 29 00      AND $200
F03C
NR ADDR UPTO RWX RANGE
00 D01F
BR 0
X

```

Figure 7.2: Example showing the Breakpoint-Add (BA) command, followed by the Walktrace (W) command and the Breakpoint-Remove (BR) command.

memory as characters, move to an empty line and enter "I <address>" and then press enter to show 32 subsequent bytes as ASCII.

7.3.3 Debugging

Debugging consists of observing the execution of a running program, interrupting it when a specific event occurs, inspecting its internal state, optionally modifying memory or instructions, and then continuing execution. CCS64 provides a set of breakpoint and walktrace commands that support this workflow while preserving the complete state of the emulated machine. Unlike many traditional machine-code monitors, CCS64 breakpoints can be triggered not only by instruction execution but also by memory reads and writes, making it possible to observe how programs interact with hardware registers and data structures.

To add automatic surveillance of various events of the processor's execution in regards of memory access, move to an empty line and enter "BA <address>" and then press [Enter] to add a new breakpoint that will be triggered on either reading (R), writing (W) or executing (X) that memory address. The BA command also supports specifying breakpoints that cover memory intervals, triggering on a specified subset of RWX and/or an interval of values of the accessed memory. To list the current active breakpoints, use the B command. An active breakpoint can be removed by issuing the command "BR <index>" which removed the breakpoint with the specified index.

The processor's execution of machine code instructions can also be single-stepped using the walktrace command, both forward (W) and backwards (WB). The walktrace commands run until the [Escape] key is pressed, and the cursor keys can be used to single-step either backwards or forwards. The vertical cursor keys step on the level of an instruction and the horizontal cursor keys step on the level of a clock cycle or microcode step.

See Figure 7.2 for an example showing a session that utilizes the breakpoint and walktrace commands in order to modify a game (i.e., training) to make the player invincible. First a breakpoint is added that triggers when the processor is reading from address D01F (I/O register that reports collisions between sprites and background). Many games read this

Chapter 8

Command-Line Reference

To get a syntax description of the current available command-line options, just run the program from a command-window with the argument "?". Example:

```
CCS64 ?
```

This will probably give an output like:

```
CCS64 [filename.{prg/p00/p01/t64/d64/g41/tap/crt}[,index]] [-cfg filename]
[-fastload] [-normalload] [-autorun] [-manualrun] [-window] [-hardsid id]
```

Example:

```
CCS64 C:\TheBestGames.D64,2 -window -autorun
```

- This will run CCS64 in windowed mode, and select the disk file image named TheBestGames.D64 as the current inserted floppy disk into disk drive with device 8, and then load the 2nd file from the directory of that floppy disk, and automatically start it.

Option	Explanation
[]	means that this option is optional and has not to be given.
{opt1/opt2/...}	means one of the options.
filename.{prg/ p00/ p01/ t64/ d64/ g41/ tap/ crt}[,index]	A file-name that describes a disk/tape/cartridge image to be loaded at the start of CCS64. The optional index tells which individual file inside of a disk/tape images that should be loaded (default is the first one). Example: mygame.d64,3
-cfg filename	Change the file-name of the config file to be used (default is c64.cfg).
-fastload	Turns on the fast-load feature when loading from disk or tape file images. Can be less accurate.
-normalload	Turns on the most accurate emulation of the disk drive when loading from disk or tape file images.
-autorun	Automatically starts the program that are loaded at the start of CCS64.
-manualrun	Do not start the program that are loaded at the start of CCS64.
-window	Force the emulator to use the windowed mode of the DirectX graphics display.
-hardsid id	Select the correct id for the HardSID sound card.

Table 8.1: Command-Line Options

Chapter 9

Configuration Reference

To change your configuration you could either use the program menu or edit the configuration file C64.CFG manually.

Parameter	Description	Values	Default
\$RGBCOLOR0	The base values for emulating the 16 different VIC-II colors.	RGB color.	0x191d19
\$RGBCOLOR1			0xfcfc9fc
...			
\$RGBCOLOR15	The screen mode ¹ which the emulator should use at startup.	Screen modes are detected dynamically and are hardware/driver dependent.	0 - Windowed (1x) mode
\$SCREENMODE			
\$WINDOWSIZEX			320 - 432
\$WINDOWSIZEY	The width of the window in pixels, when CCS64 is run in Windowed (1x) mode.	200 - 306	384
\$SAMPLEBITS	The height of the window in pixels, when CCS64 is run in Windowed (1x) mode.	16 - 16 bits samples. 8 - 8 bits samples	282
\$SAMPLEFREQ	The number of bits to use for the samples created by the SID emulation for sound output.	44100 or any other supported value.	16
\$SOUNDENABLE	The frequency of the samples created by the SID emulation for sound output.	YES - Sound output on. NO - Sound output off	44100
\$LOADPATH	If sound output should be enabled or not.	LEFT - Use the left shift key. RIGHT - Use the right shift key.	YES
\$SHIFTKEYAUTO	The path where you have your C64 games and demos.		
	Which shift key to use to emulate the emulate the C64 cursor up/left keys.		

Table 9.1: Caption

Chapter 10

Troubleshooting

- **Compatibility Problems when Loading from D64.** Sometimes special games on .D64 images will behave badly, when you use the 'Run (Fast)' or 'Boot (Fast)' feature (this is default). To solve these kind of problems, use the 'Run (Normal)' or 'Boot (Normal)' features instead; otherwise, perform the following:
 - 1) Re-set the C64.
 - 2) Enter the "1541 Device 8 ..." menu, high-light the correct .D64, then select the "Insert Disk" option.
 - 3) From the BASIC type:
`LOAD"*,8,1`
- **Device Problems in Menu.** If you do not have a CD-ROM in your CD-ROM drive or a floppy disk in your floppy disk drive, the emulator will hang, if you select this device in the menu. So please, always be sure that you have an initialized (sometimes you have to wait some seconds after instering it) disk in your drive, before trying to use that device from within the menu.
- **Turbo Tape.** The option Load As Turbo-Tape should only be used when you get a message like "PRESS PLAY ON TAPE" on your C64-screen, and you're sure about that the second part should be loaded as turbo-tape. Otherwise, nothing useful will happen.
- **Sound Sampling Quality.** Note that when your sampling the C64's audio output to a .WAV file, the sample rate and bit resolution will be the same as selected in the C64.CFG file.
- **German Keyboard.** Note that on a German keyboard, the keys [Y] and [Z] will have swapped, when using under CCS64.
- **Screen Shot File Names.** When you use the Screen-Shot function, the image will be saved as CCS0.BMP, CCS1.BMP, CCS2.BMP, and so on.
- **Sound Sampling File Names.** When you use the Sound Sampling function, the audio file will be saved as CCS0.WAV, CCS1.WAV, CCS2.WAV, and so on.

- **Video Sampling File Names.** When you use the Video Sampling function, the video file will be saved as CCS0.AVI, CCS1.AVI, CCS2.AVI, and so on. NOTE: AVI files are un-compressed, i.e. very large; therefore they should be re-compressed, using video editing software, prior to further distribution.
- **Printer Output.** When you use the printer emulation, the output will be saved as PRINTER4.TXT, PRINTER5.TXT, and so on.
- **HardSID Configuration.** When you use the HardSID support, the "Channels 1-4 Burst" and "Channels 1-4 Buffered" modes are recommended for faster CPUs ; the "Channels 1-3 Burst & Channel 4 Emulated" mode is recommended for slower CPUs.
- **Configuration Problems.** If you have some problems with the configuration of CCS64, and want to go back to the factory-default settings, just delete the C64.CFG file from within the CCS64 installation directory.
- **Sound Problems.** Using the emulator under Windows with old and out-dated sound drivers can sometimes cause the system to hang or for there to be no sound at all.

Chapter 11

Technical Notes

The C64 and 1541 ROMs are (C) Commodore Business Machines. If you don't want these original ROMs, you should replace the files KERNAL.ROM, BASIC.ROM, C1541.ROM, and CHAR.ROM with alternative ROM images.

PAL TV palette emulation by John 'Graham' Selck and Philip 'Pepto' Timmermann.

SID filter emulation contributed by Michael Kleps.

CCS64 Keyboard Layout (PDF) document by Stilianos 'Stello' Doussis.

Chapter 12

Contact / Project Links

If you have any emulation problems or suggestions, please contact me. Note that I get a lot of e-mail, so you cannot count on me reply to your message. But, of course, I will read it.

E-Mail: phs@ccs64.com

CCS64 Homepage: <http://www.ccs64.com>

Computerbrains Homepage: <http://www.computerbrains.com>

Note that these addresses may change in the future, so you can find my new addresses by searching the Internet.

This program is released as SHAREWARE and is therefore freely distributable, although you may not charge anyone for more than the actual cost of the distribution. The meaning of SHAREWARE is that, if you use a program regularly, you should strongly consider registering it. When you register a program, it means that you are actually buying it.

The installation and use of this product is done completely under your own responsibility. Under no circumstances should the producers of this product be held responsible for any damages or costs that this product in any way could cause you or your equipment.

As this is SHAREWARE, if you like and use this program regularly, you should register it. At the moment the register fee is 30 USD, using PayPal <https://www.paypal.com/paypalme/ccs64/30usd>.